# AUTHENTICATION AND ENCRYPTION FOR RADNET

## *Introduction*

We propose to transfer cryptography technology that is operational and approved by the IAEA to the RadNet standard. This transfer provides for a well proven implementation with minimal development. The IAEA approval transfers the benefit of 3rd-party vulnerability assessments and the confidence in the implementation that those assessments provide.

We propose that both authentication and encryption be implemented in a way that is 1) exportable, and 2) minimizes the complexity of the key management system. Both of these implementation features are extremely important to the legal delivery of the systems abroad, and to make such systems administratively feasible to organizations like the IAEA.

## *Background*

The nomenclature that describes key use, generation, and distribution; and their use to provide digital signatures and encryption is more than a little confusing. First, secret keys are just that. They must be held secret or all security is lost. Secret key algorithms are symmetric, that is, the encryption key is the same as the decryption key. If a secret key is ever released, all the data encrypted with that key is compromised. Managing secret keys requires vaults with secure computers and human delivery of keys. Public-key cryptography is very different. Keys come in pairs: a private key that is kept secret and a public key that can be posted on bulletin boards and the internet. The public-private pair is not arbitrary; they are relatively prime with respect to modular exponentiation and need not be the same length. The algorithms are not symmetric. Items encrypted with one member of a pair can only be decrypted with the other member.

It is important to note that public-key authentication is transparent to data handling and imposes very small administrative burdens for key-management. Encryption is exactly the opposite. There is no possible way to make key-management transparent; in fact, the concept is an oxymoron. The data messages are being inexorably scrambled by the encryption. Any loss of keys, confusion in the distribution, or change of keys will absolutely make the data involved totally unrecoverable. If this were not the case, the encryption would be useless. Therefore, significant attention must be paid to key management and distribution in order to preserve the long-term integrity of the data.

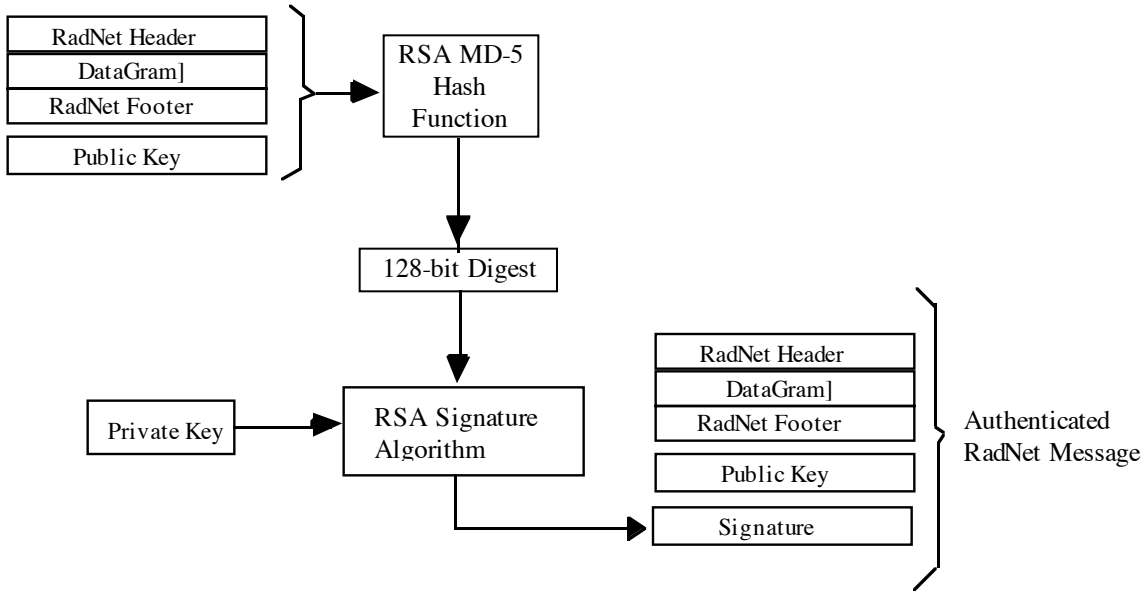
The methods we propose are based on public-key cryptography because it significantly eases the key management burden when compared to other options.

### Signing and Signature Verification (Authentication)

The objective of authentication is to allow anyone to verify the signature and thus be sure that the attached data (message) is valid and has remained unaltered since the signature was attached. A digital signature does not hide data from anyone; it is still plaintext. Consequently, the entire original message is available for use by other applications as if the signature were not present. To the extent that the original document contains information that verifies the source of the data, then the digital signature also certifies that the data originated from a specific sensor.

The signature is provided in two steps. In the first step, the transmitter's *public* key is appended to the message and the resulting combination is processed through a secure hash function such as MD5. These functions are available from certified libraries and produce a

128-bit representation (digest) of the input that is sensitive to a change of only one bit in the entire input. The second step is to encrypt the digest with the transmitter's *private* key to produce the signature. The signature is then appended to the compound structure to form an authenticated message.




Because the public key is appended to the signed document, any receiver can verify the authenticity of the message without access to the transmitter's private key. Furthermore, since no one has access to the private key, the signature cannot be forged.

The signature is verified by repeating the signing process. That is, the signature is decrypted with the public key. Then the compound message without the signature is submitted to the same hash function as the transmitter's. If the resulting digest matches the decrypted signature, the document is valid. Verification of messages can be centralized in a server before distribution to the message processing clients; or each client can verify the signatures. The choice of implementation would depend upon the system as a whole and the actual end use. The signing and verification process remain unchanged.

In the RadNet domain, each UDP packet of a compound message would be independently signed.

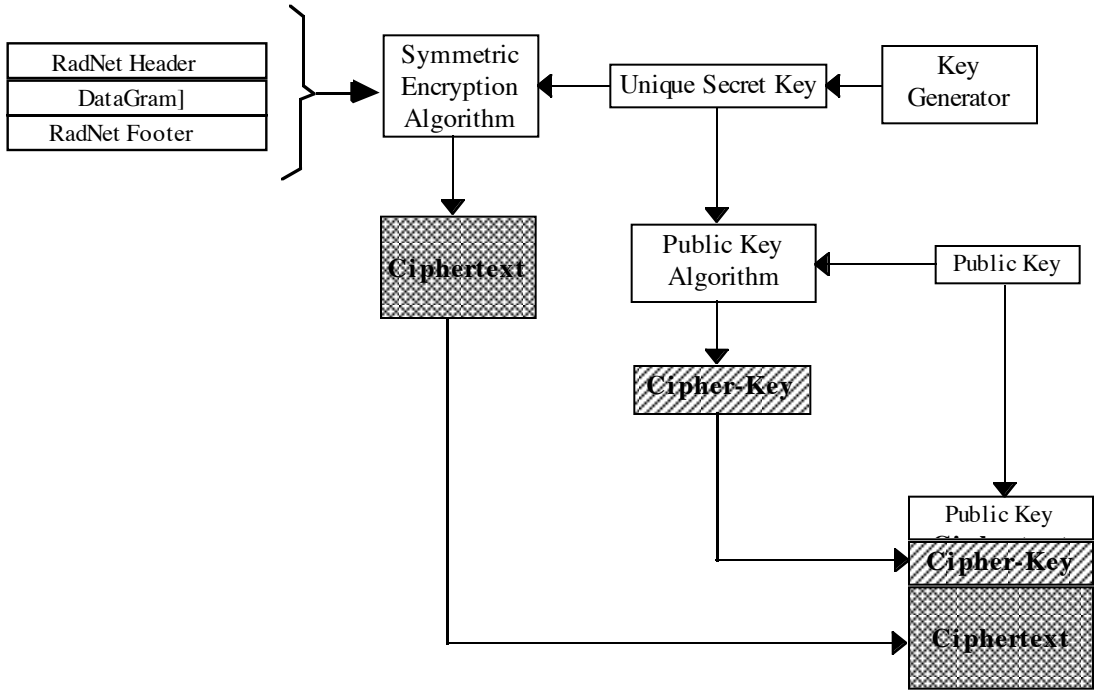
### Encryption and Decryption

The encryption process is intended to prevent any unauthorized person from viewing the data. However, encryption does not necessarily authenticate the data. Encryption introduces a complication into the normal flow of data. Any part of the "message" that is encrypted cannot be read by any application that does not decrypt it first. This can become problematic if services are used to sort, route, store, or otherwise operate on the data or the packet header. Consequently, encryption is not transparent as is authentication.

Conceptually, encryption operates like authentication run backwards. The *recipient* of the data generates the key pairs, keeps the private key secret, and distributes the public key to every device that will send data to the recipient. (In principle, there could be a key-pair for every device) The public keys can be distributed over the Internet if desired. Each device uses the public key for encryption. Once data is encrypted with the public key, only the holder of the corresponding private key (the desired recipient) can decrypt and read the data.

Because anyone with the public key can send data to the recipient, it is possible to "forge" data. Therefore, it is always advisable that data be signed (authenticated) before it is encrypted; and furthermore, that the keys used for encryption and authentication be different.

Unfortunately, there is a pragmatic problem with encryption. Secret-key methods are fast; but impose a nearly impossible key management burden on the user. Public-key methods reduce the key management problem but are very slow and compute-intensive. In general practice, a combination of the two methods is used. As shown at the top of the figure below, each message is encrypted using a secret-key algorithm with a key that is uniquely generated for each data package. Once the ciphertext is produced, the unique secret key is encrypted using public key methods to produce a "cipher-key" The encrypted secret key and the public key are pre-pended to the encrypted message to produce a "packet". In common practice, this technique is known as an "RSA envelope."




The result is that the data is encrypted using a fast algorithm and the key for decrypting it is distributed with the data in a way that can be recovered using the more complex public-key algorithm. Further, each data object uses a unique key, so that an attacker would have to cryptanalyze every data item independently; i.e. the key from one data item would have no value for the next.

## PROPOSED IMPLEMENTATION

### *Overview*

In the sections that follow we propose methods for key generation at both the server and the transmitting instrument, for signing the messages at the instrument and verifying them at the server (authentication), and for encrypting signed messages at the instrument and decrypting them at the server.

### *Key Generation*

When using public-key methods it is not necessary to change keys frequently . In fact, rigorous control of key change provides a simple but effective registry and obviates the need for complex key management.

#### Transmitting Instrument

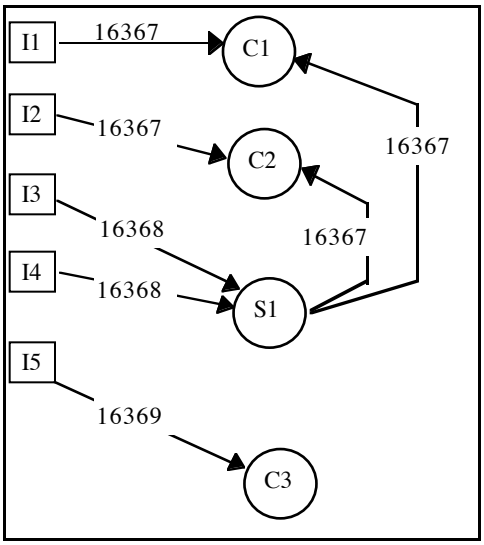
Keys must be generated within the transmitting instrument using random numbers that are "sufficiently random for cryptography." This precludes use of typical pseudo-random number generators. Further, the key-pair must be stored in non-volatile memory so that the keys survive power cycling. Finally, the private key must never be accessible in a manner that would allow an adversary access to the key. We recommend that key generation be somewhat cumbersome and require some direct hardware connection, such as an internal jumper, so that a key change becomes direct evidence of a tamper.

#### Receiving Server

There are many options for key generation within the server. The same randomness criterion applies to the server as to the instrument. Further, the same security for the private keys applies to the server as to the instruments. Key generation should be somewhat cumbersome so that it cannot happen by mistake, but not nearly so restrictive as for the instruments. A standard software key generator from the registered cryptography library can be used.

### *Authentication*

The intent of the application of authentication not interfere with clients that do not use authentication. Second, the proposed application is general enough that actual use can adapt to system-level requirements in any facility, as shown in the diagram. Here, we have two instruments connecting directly to clients without the use of authentication. We have two others that connect to the same clients through a service that verifies the authentication. The last instrument connects to a different client that verifies the authentication internally.




#### Transmitting Instrument

The instrument's private key is used only for authentication. The corresponding public-key is the transmitting instrument's certified signature of legitimacy.

The secure hash function that is used to generate the message digest must be both well known to be cryptographically secure and the mode of operation (padding, salting, CBF, etc.) must be precisely known by both the transmitting instrument and the receiving client.

Encryption of the message digest must be performed using the instrument's private key and an algorithm that is known to be cryptographically secure.

Both the hash function and the encryption algorithm should be implemented using certified cryptographic libraries..

### Receiving Server

The server must validate the signature on the data from the transmitting instrument; either as a general service or as contained within the client software specific to the instrument.

Because the key needed to verify the authenticity of the data is contained within the data, the verification process can be handled in software using a certified library. It is necessary that the verification process know the details of the implementation of the secure hash function in the transmitting instrument.

### Proposed Data Structure

The existing RadNet data structure must not be disrupted by adding authentication. There are many applications which do not require authentication, and these must not be required by the standard to implement unnecessary functionality.

We propose that 3 unassigned bytes in the header be designated for use by all cryptography functions, including authentication and encryption. Values in this byte can indicate both the presence of cryptographic functions and their type.

We propose that the cryptographic information always be appended to the end of a message so that it does not interfere with existing structures. This approach allows the data-users of RadNet to be completely unconcerned with the cryptography, and allows the cryptography to adapt easily to any additions to the RadNet structure.

The authentication process will use two of the 3 reserved bytes in the message header to contain the message length needed for generating the digest. The third byte will be used as a flag to indicate the verification status.

Authentication requires the following additional data-space at the end of the RadNet message:

64 bytes for the message digest

96 bytes for the public key

Total authentication space -- 160 bytes.

| | | |
|---|---|---|
| Original message with flag in header | Variable | Indicates authenticated message |
| Public key | 96 bytes | Transmitting instrument's public key |
| Digest | 64 bytes | Signature |

It is proposed that the RadNet message standard reserve 300 bytes for the cryptographic information. This provides flexibility for longer keys, different algorithmic data, and larger message digests in the future.

## *Encryption*

The use of encryption must be optional within the RadNet Standard.

Although prudent cryptographic practice dictates that encryption never be implemented in the absence of authentication; the RadNet Standard need not specify such implementation.

Because encryption makes the message totally unrecoverable in the event of a lost or disrupted key, and because key management is critical to the security of the system as a whole, we propose that encryption be implemented as a public-key envelope containing the message. With encryption applied as an envelope, the envelope must be processed before the contents can be exposed for use by other processing.

### Transmitting Instrument

Encryption is performed in the transmitting instrument using the public key of the connected server. This implies that the implementation includes a method for delivering key to the instrument and that the instrument will store the server key in non-volatile storage.

We propose that encryption be performed entirely in software using a registered library. The entire signed message will be encrypted using a random key and a symmetric algorithm. The symmetric key will be encrypted using the receiver's public key. A message header, the encrypted symmetric key, and the receiver's public key will be *pre-pended* to the encrypted message. The message header will constitute a new message type and will consist of 3 bytes of all 1's. This unique header will signify to the receiver and to clients that the message is encrypted.

### Receiving Server

The service that is accepting incoming messages will recognize the unique header for encryption and route the message in accordance with the rules for the installation

### Proposed Data Structure

We recommend the following header structure to accommodate key registration in addition to the basic function.

| | | |
|---|---|---|
| Header | 3 bytes | All >FF Indicates encrypted message |
| Crypto-data | 24 bytes | Implementation dependent. |
| Public key | 96 bytes | Server's public key |
| Secret key | 5 bytes | Encrypted secret key |
| Body | Variable | Encrypted message |

## PROPOSED ADDITION TO THE RADNET STANDARD TO ACCOMMODATE AUTHENTICATION AND ENCRYPTION

### *Authentication*

#### General

Authentication requires that the message length be known in order that the digest can be reliably reproduced. Therefore, it is necessary that the message header contain that information. In addition, it is necessary for the server-mode implementation that the clients know if the message is valid or if it failed verification. This allows each client to take application-specific action upon authentication failure.

#### Message Header

Reserved bytes 50, 51, and 52 are proposed to be used by the authentication process.

| | |
|---|---|
| Bytes 50, 51 | The length in bytes of the original message. If non-zero, indicates that authentication is in effect. If zero, then authentication is not implemented |
| Byte 52 | "Invalid" flag. This byte is always set to zero when the message is transmitted. Authentication services set this byte to a non-zero value if the message fails signature verification. Clients check this byte with zero meaning valid data; and take appropriate "bad data" action if the byte is non-zero. |

#### Message Structure

The RadNet message reserves 300 bytes at the end of the actual instrument message for the authentication signature information. Exactly how these 300 bytes are used depends upon the specifics of the signature algorithm and the size of the keys used in the application. Because the verification process must be exactly consistent with the signing process, there is no ambiguity. Different algorithms will use different structures, but there is no opportunity for confusion since there must be no ambiguity in the pairing process.

### *Encryption*

#### General

Because encryption hides all of the information in the RadNet packet, it is necessary to create a special packet-type that signals the general receiving processes that the packet is encrypted. Use of a normally-invalid header code will also cause a client that receives an encrypted message in error to reject the message.

#### Message Header

| | |
|---|---|
| Bytes 1 to 3 | Set to all ones (>FF) to indicate that the message is encrypted. |
| Bytes 4 to 128 | Various crypto-data as needed, including keys and pointers. |
| Bytes 129 to end | Encrypted message |